LOGGING AND ANALYZING COMPUTER USER'S CONTEXT DATA

TECHNICAL FIELD

The present invention is directed to the field of data storage and analysis, and, more particularly, to the field of storage and analysis of user context data, such as in a wearable personal computer.

CROSS-REFERENCE TO RELATED APPLICATIONS

[illegible] 1/21/03

This application is a continuation-in-part of U.S. Patent Application No. 09/216,193, and now US Patent No. 6466232 entitled "METHOD AND SYSTEM FOR CONTROLLING PRESENTATION OF INFORMATION TO A USER BASED ON THE USER'S CONDITION" filed December 18, 1998, and a continuation-in-part of U.S. Patent Application No 09/464,659, and now US Patent No. 6513046 entitled "STORING AND RECALLING INFORMATION TO AUGMENT HUMAN MEMORIES" filed December 15, 1999. These applications are hereby incorporated by reference in their entirety.

BACKGROUND

Most general-purpose computer systems are designed to operate by processing commands as they are inputted. As a result, the only information such a computer system typically possesses about the condition of its user is the commands issued by the user. In particular, such a computer system generally only possesses this very limited type of information about the user's condition for the short time during which the computer system is processing the command.

A few special-purpose application programs that execute on general-purpose computer systems maintain a record of the commands issued by the user. For example, the Microsoft Internet Explorer and Netscape Navigator web browser applications accept commands to display particular web pages, and have a "history" feature that maintains and displays a list of web page requests made by the user. The history feature enables the user to review his or her web page requests, and to resubmit a web page request by selecting it from the list.

While such application history features can be useful, they are not extensively configurable, with respect to either the identity of the information that they record, the manner in which they record it, the manner in which they maintain it once recorded, or the nature of any later-performed analysis on the recorded information. Further, such history features are inherently limited to recording information in the possession of their host applications, and thus do not record other information such as the state of other applications, the operating system, or the computer system hardware. Such history features also fail to record information about the user, or about the environment surrounding the computer system and/or the user.

Accordingly, a facility in a general-purpose computer system for selectively recording, maintaining, and analyzing information about the computer system, a user, and/or their environment would have significant utility.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 illustrates an embodiment of the facility which executes on a general-purpose body-mounted wearable computer worn by a user.

Figure 2 illustrates an exemplary computer system on which an embodiment of the facility is executing.

Figure 3 is a data flow diagram showing a sample exchange of attributes performed by the characterization module.

Figure 4 is a data structure diagram showing an attribute instance table in which the characterization module caches attribute instance values.

Figure 5 is a data structure diagram showing a logging configuration table used by the facility to determine which attribute instances to log and when.

Figure 6 is a flow diagram showing the steps preferably performed by the facility in order to log attribute instances.

Figure 7 is a data structure diagram showing an attribute log table produced by the facility.

Figure 8 is a flow diagram showing the steps preferably performed by the facility in order to maintain an attribute log.

Figure 9 is a flow diagram showing the steps preferably performed by the facility to analyze log records for attribute instances.

Figure 10 is a flow diagram showing the steps preferably performed by the facility in order to conduct a context simulation.

Figure 11 is a data flow diagram showing the operation of the facility without a characterization module.

DETAILED DESCRIPTION

A software facility for recording, maintaining, and analyzing data regarding user context ("the facility") is provided. In a preferred embodiment, a characterization module operating in a computer system such as a wearable general-purpose computer system obtains context information, in the form of values for individual attributes each modeling an aspect of the wearable computer system, its user, or the surrounding environment, from one or more context servers, and provide it to one or more context clients that consume the context information. The facility, preferably implemented as part of the characterization module, periodically stores values of selected attributes obtained from context servers in a time-indexed log.

The log may be stored in a database, a sparse matrix, or a file of another type, on a storage device that is either local or remote with respect to the computer system. The identity of the logged attributes, as well as the frequency at which each is logged, is preferably configurable, either manually by the user, or by other programs executing on the computer system. The logged attributes are data gathered or derived to characterize the state of the computer system, its user, and the surrounding environment. The logged attributes may be very compact, such as a boolean value, or very voluminous, such as a color image.

The facility preferably further provides configurable pre-logging processing of logged attributes, such as determining whether data specified for logging should actually be logged, transforming data into a new form for logging, and/or combining data from different sources before loggging it.

The facility preferably further provides configurable, selective maintenance of logged attribute data, enabling logged data to be compacted, abstracted, or otherwise further processed based on a variety of factors. This enables the facility to retain detailed, voluminous information for a short time, then process the information to reduce the volume of information that is retained for a longer time. In cases where the detailed information is interesting, such processing can be skipped or deferred to retain the detailed information.

The facility preferably further provides analysis of logged attribute data using later-defined analysis techniques. For example, after data has been logged indicating that (1) a user was present in a particular location for a period of time, and that (2) the user thereafter moved away from that location at a particular direction and speed, an analysis technique could be defined and applied to this data to determine that the user waited for and embarked on a mode of transportation that takes on passengers at the logged location and moves in the logged direction at the logged speed.

Additional embodiments of the facility provide retrospective simulation of events represented in logged attribute data by substituting logged values of selected attributes for real-time attribute values in the attribute values provided to context clients by the characterization module. Such simulations may be conducted for a variety of purposes, including for training, to demonstrate how context clients perform in a particular situation, for usability texting, for debugging, or to repeat an experience represented by logged data. By modifying the logged data used in a simulation beforehand, the user can modify the represented experience. Indeed, completely new experiences may be presented to the user by manufacturing logged data that represent such new experiences.

In this way, the facility makes it possible to preserve, review, and analyze, and simulate historical context data. This enables programs executing on the computer system to make observations, and even judgments, in the present that are predicated on an understanding of events that occurred in the past, and to take initiative to assist the user on this basis. It also enables programs executing on the computer system to evaluate and understand in a retrospective context, rather than merely in an instantaneous context.

Figure 1 illustrates an embodiment of the facility 100 which executes on a general-purpose body-mounted wearable computer 120 worn by a user 110. Many wearable computers are designed to act as constant companions and intelligent assistants to a user, and are often strapped to a user's body or mounted in a holster. The computer system may also be incorporated in the user's clothing, be implanted in the user, follow the user, or otherwise remain in the user's presence. In one preferred embodiment the user is human, but in additional preferred embodiments, the user may be an animal, a robot, a car, a bus, or another entity whose context is to be logged. Indeed, the computer system may have no identifiable user, but rather operate as an independent probe, logging and/or reporting on the context in an arbitrary location.

The wearable computer 120 has a variety of user-worn user input devices including a microphone 124, a hand-held flat panel display 130 with character recognition capabilities, and various other user input devices 122. Similarly, the computer has a variety of user-worn output devices that include the hand-held flat panel display, an earpiece speaker 132, an eyeglass-mounted display 134, and a tactile display 136. In addition to the various user-worn user input devices, the computer can also receive information from various user sensor input devices 116 and from environment sensor input devices 128, including video camera 121. The characterization module can receive and process the various input information received by the computer, such as from context servers that process the input information and generate attribute values, and can present information to the user on the various output devices accessible to the computer.

In the current environment, computer 120 is accessible to a computer 150 (*e.g.*, by being in wireless proximity or by being reachable via a long-distance communication device such as a cellular phone) which also has a variety of input and output devices. In the illustrated embodiment the computer 150 is non-portable, although the body-mounted computer of the user can similarly communicate with a variety of other types of computers, including body-mounted computers of other users. The devices from which the non-portable computer can directly receive information include various user input devices 152 and various user sensor input devices 156. The non-portable computer can output information directly to a display 160, a speaker 162, an olfactory device 164, and a printer 166. In the illustrated embodiment, the body-mounted computer can communicate with the non-portable computer via a wireless transmission medium. In this manner, the characterization module can receive information from the user input devices 152 and the user sensor devices 156 after the information has been transmitted to the non-portable computer and then to the body-mounted computer. Alternately, the body-mounted computer may be able to directly communicate with the user input devices 152 and the user sensor devices 156, as well as with other various remote environment sensor input devices 158, without the intervention of the non-portable computer 150. Similarly, the body-mounted computer may be able to supply output information to the display 160, the speaker 162, the olfactory device 164, and the printer 166, either directly or via the non-portable computer, and directly to the telephone 168. As the user moves out of range of the remote input and output devices, the characterization module will be updated to reflect that the remote devices are not currently available.

The various input devices allow the characterization module or an application such as a context server (not shown) executing on the computer system 120 to monitor the user and the environment and to model their current condition. Such a model can be used by various applications, such as context clients, for various purposes. A model of the current conditions can include a variety of condition variables that represent information about the user, the computer, and the user's environment at varying levels of abstraction. For example, information about the user at a low level of abstraction can include raw physiological data (*e.g.*, heart rate and EKG) and geographic information (*e.g.*, location and speed), while higher levels of abstraction may attempt to characterize or predict the user's physical activity (*e.g.*, jogging or talking on a phone), emotional state (*e.g.*, angry or puzzled), desired output behavior for different types of information (*e.g.*, to present private family information so that it is perceivable only to myself and my family members), and cognitive load (*i.e.*, the amount of attention required for the user's current activities). Background information which changes rarely or not at all can also be included, such as the user's age, gender and visual acuity. The model can similarly hold environment information at a low level of abstraction, such as air temperature or raw data from a motion sensor, or at higher levels of abstraction, such as the number and identities of nearby people, objects, and locations. The model of the current conditions can additionally include information added explicitly from other sources (*e.g.*, application programs), as well as user-specified or system-learned defaults and preference information.

Those skilled in the art will appreciate that computer systems 120 and 150, as well as their various input and output devices, are merely illustrative and are not intended to limit the scope of the present invention. The computer systems may contain additional components or may lack some illustrated components. For example, it is possible that the characterization module can be implemented on the non-portable computer, with the body-mounted computer replaced by a thin context client such as a transmitter/receiver for relaying information between the body-mounted input and output devices and the non-portable computer. Alternately, the user may not wear any devices or computers.

In addition, the body-mounted computer may be connected to one or more networks of other devices through wired or wireless communication means (*e.g.*, wireless RF, a cellular phone or modem, infrared, physical cable, a docking station, etc.), either with or without support from other computers such as the computer 150. For example, the body-mounted computer of a user can make use of output devices in a smart room, such as a television and stereo when the user is at home, if the body-mounted computer can transmit information to those devices via a wireless or wired medium or if a cabled or docking mechanism is available. Alternately, kiosks or other information devices can be installed at various locations (*e.g.*, in airports or at tourist spots) to transmit relevant information to body-mounted computers within the range of the information device.

Those skilled in the art will also appreciate that specialized versions of the body-mounted computer and characterization module can be created for a variety of purposes.

Figure 2 illustrates an exemplary computer system 200 on which an embodiment of the facility is executing. The computer includes a memory 230, a CPU 210, a persistent storage device 250 such as a hard drive, and input/output devices including a microphone 222, a video camera 223, a computer-readable media drive 224, such as a CD-ROM drive, a visual display 225, a speaker 226, and other devices 228. The memory preferably contains the facility 231, as well as information reflecting the current state of the facility (facility state) 232. The memory further contains characterization module 233, software modules 234, 235, and 238 that consume attributes and are therefore context clients, and software modules 236, 237, and 238 which provide attributes and are therefore context servers. While items 231-238 are preferably stored in memory while being used, those skilled in the art will appreciate that these items, or portions of them, can be transferred between memory and the persistent storage device for purposes of memory management and data integrity.

Attributes represent measures of specific context elements such as ambient temperature, location and current user task. Each attribute preferably has the following properties: a name, a value, an uncertainty level, units, and a time stamp. Attributes provided through the characterization module by a context server may either be "measured," in that they are directly received from an input device, or "derived," in that they are the result of performing processing on values directly obtained from input devices other attributes. Indeed, a derived attribute may be produced by performing additional processing on other derived attributes. Context servers, in addition to providing attributes through the characterization module, may also provide other functions. For example, an application, such as an electronic mail application, may serve as a context server by providing attributes through the characterization module. In addition to the source of attributes described above, such an "expanded" context server may provide attributes relating to the other functions of the expanded context server. For example, an electronic mail application context server could provide an attribute indicating other new messages are waiting. Indeed, the same program module may operate both as a context client and a context server.

Two or more different context servers may preferably supply to the characterization module their own values for a single attribute. For example, a first context server can supply a value for a user.location attribute based on data received from a global positioning system device, while a second context server can supply a value for the user.location attribute based on data received from an indoor positioning device. These alternate values for the same attribute provided by different context servers are referred to as separate "instances" of the attribute. Separate instances of an attribute provided by different context servers may be based on data obtained from the same sensor, or may be based on data obtained from different sensors. The characterization module preferably provides a variety of different approaches, called "mediators," for determining, when an attribute requested by a context client has more than one instance, what attribute value to provide in response to the attribute request.

When the characterization module obtains an attribute value from a context server, it preferably caches it for responding to future requests for the attribute from context clients. Such attribute requests may specify a specific instance of the attribute—that is, a specific context server from which the attribute is to be obtained—or may specify that the attribute is to be obtained by applying a particular mediator to whatever instances of the attribute are available, or may utilize a default mediator to mediate between available instances of the attribute. When the characterization module receives an attribute request from a context client, it identifies the attribute instances available to satisfy the request, and, for each, determines whether the value for the attribute instance cached by the characterization module was obtained sufficiently recently from the corresponding context server. If not, the characterization module requests an updated value for the attribute instance from the corresponding context server before performing any necessary mediation and returning a value to the context client.

The facility preferably supports the collaborative logging, in which attribute values produced in multiple computer systems are logged in a single consolidating computer system and there analyzed, or in which values logged in multiple computer systems are maintained, analyzed, or simulationed in a single consolidating computer system. Such collaborative logging is valuable where multiple computer systems are collecting and processing information about the same general subject. For example, information about a military battle collected and processed by separate computer systems carried by each of a number of soldiers may preferably by consolidated and analyzed by the facility in a strategist's computer system.

The characterization module preferably utilizes a plain-language, hierarchical, taxonometric attribute nomenclature to name attributes. The attribute names within the nomenclature are preferably specific so that there is no ambiguity as to what they represent. The characterization module preferably supports the extension of the nomenclature by adding new attribute names that conform to the hierarchical taxonomy of the nomenclature. The nomenclature preferably has attribute names relating to the user, such as user.position, user.speed, and user.direction, providing information about the user's position, speed, and direction, respectively. The nomenclature preferably contains attribute names for various user moods, or "affect," such as user.mood.happiness, user.mood.anger, and user.mood.confusion. The nomenclature preferably includes attribute names for user activities, such as user.activity.driving, user.activity.eating, and user.activity.sleeping. The nomenclature preferably includes attribute names for user physiology values, such as user.physiology.pulse, user.physiology.body_temperature, and user.physiology.blood_pressure. The nomenclature preferably includes attribute names for similar attributes of people other than the user, such as person.John_Smith.mood.happiness. The nomenclature preferably includes attribute names for aspects of the computer system or "platform." For example, the nomenclature preferably includes attribute names for aspects of the platform's user interface, such as platform.user_interface.oral_input_device_availability and platform.user_interface.visual_output_device_availability. The nomenclature preferably includes attribute names for attributes relating to the central processing unit, or "CPU," of the platform, such as platform.cpu_load and platform.clock_speed. The nomenclature preferably also provides attribute names for aspects of the local environment, such as environment.local.time, environment.local.temperature, and environment.local.ambient_noise_level. The nomenclature preferably also includes attribute names for remote environments, such as environment.place.chicago.time and environment.place.san_diego.temperature. The nomenclature preferably further provides attribute names relating to specific applications. For example, the nomenclature preferably provides attribute names for aspects of an electronic mail application, such as application.mail.available, application.mail.new_messages_waiting, and application.mail.messages_waiting_to_be_sent. The nomenclature preferably further provides attribute names relating to data stored in or accessible to the computer system. The nomenclature preferably further provides attribute names relating to the connectivity of the computer system to other computer systems and similar devices, such as such as connectivity.Internet.available and connectivity.Internet.speed.

A single attribute may preferably be referred to by two or more names in the nomenclature. For example, the attribute names person.John_Smith.mood.happiness and user.mood.happiness may refer to the same attributes in a computer system whose user is John Smith.

In this manner, the attribute nomenclature used by the characterization module provides effective names for attributes relating to the user, the computer system, and the environment. Additional detail on the attribute nomenclature utilized by the facility is provided in U.S. Patent Application No. 09/724902, entitled "Dynamically Exchanging Computer User's Context," which is hereby incorporated by reference in its entirety.

1/27/03 DM

Additional embodiments of the facility preferably identify attributes in a variety of ways other than the plain-language, hierarchical, taxonometric attribute nomenclature discussed above. For example, attributes may be identified by arbitrary identifiers, or by the identity of associated events or messages.

Additional details of the operation of the characterization module to obtain attribute values from context servers is provided by U.S. Patent Application No. 09/692507, entitled "Interface for Exchanging Context Data", which is hereby incorporated by reference in its entirety.

1/27/03 DM

Figure 3 is a data flow diagram showing a sample exchange of attributes performed by the characterization module. The diagram shows characterization module 300, as well as five other software modules, 310, 320, 330, 340, and 350. Software modules 310, 320, and 330 are said to be context servers, in that they provide attributes to the characterization module. For example, context server 330 provides an user.in_region attribute 331. It can be seen that context servers may provide more than one attribute. For example, context server 320 provides a user.location attribute 321 and an user.elevation attribute 322. It can further be seen that a single attribute may be provided by more than one context server. For example, context server 310 provides user.location attribute 311, while context server 320 provides user.location attribute 321.

The diagram further shows the facility storing attribute instance values obtained from context servers by the characterization modules in a logging database 360 that is comprised of one or more attribute instance logs.

Figure 4 is a data structure diagram showing an attribute instance table in which the characterization module caches attribute instance values. The attribute instance table contains a row for each attribute instance created by a context server. The attribute instance table 400 contains rows 401-404, each corresponding to a different attribute instance. Each of these rows contains the following fields: an attribute name field 411 containing the name of the attribute, a context server name field 412 identifying the context server that created the attribute instance, a value field 413 containing the value of the attribute last provided by the context server, and uncertainty field 414 identifying the level of uncertainty of the value, a timestamp 415 indicating the time at which the value is effective, a units field 416 identifying the units for the value and the uncertainty, and an indication 417 of the number of context clients consuming the attribute instance. For example, row 401 indicates that an instance of the user.location attribute from the gps context server has the effective time of 13:11:04.023 on 2/22/2000. Its value is 47° 36.73' N, 122° 18.43' W degrees and minutes, with an uncertainty of 0° .09'. It further indicates that two context clients are presently consuming this attribute instance. It should be noted, as shown, the user's location, expressed in terms of latitude and longitude, is maintained in a single attribute. In alternative embodiments, such a location could be maintained in a larger number of attributes. For example, rather than being maintained in a single user.location attribute, such a location could be distributed across four separate attributes: user.location.latitude.degrees, user.location.latitude.minutes, user.location.longitude.degrees, and user.location.longitude.minutes. The characterization module preferably adds row 404 to the attribute instance table when the location_region_analysis context server calls the CreateAttributeInstance function for the user.in_region attribute.

Figure 5 is a data structure diagram showing a logging configuration table used by the facility to determine which attribute instances to log and when. The information stored in the logging configuration table may be provided by a variety of sources, including the user, the facility, applications, and other programs running on the computer system. In particular, the facility or another program may dynamically determine that a particular attribute instance should be logged, and accordingly add a logging configuration record to the logging configuration table. For example, a program may determine that an altitude parameter should be logged when it observes that the user's location corresponds to the location of a helipad.

The logging configuration table 500 has two rows 501 and 502. Each row corresponds to a particular attribute instance and specifies details of how that attribute instance is to be logged. Each row preferably contains a log name field 511 naming the logging which values of the attribute instance are to be stored; an attribute name field identifying the attribute name of the attribute instance to be logged; a context server name field 513 identifying the context server that is the source of the attribute instance to be logged; a logging frequency field 514 indicating the interval at which the attribute instance is to be logged; a maximum attribute value age field 515 indicating the maximum age of a cache value of the attribute instance that may be logged; an importance level field 516 indicating the importance of maintaining logged data for the attribute instance; a pre-logging processing field 517 indicating any processing to be performed on the attribute instance before storing in the log; a maintenance processing field 518 indicating any maintenance processing to be performed on log values of the attribute instance while they are stored in the log; and a last logged field 519 indicating the time at which the attribute instance was most recently logged.

For instance, row 502 indicates that the instance of the user.in_region attribute supplied by the location_region_analysis context server is to be logged in the user_log log. While only one unique log name is shown in the logging configuration table, the use of the log name field enables the facility to log different attribute instances in different logs, for instance, when the attribute instances relate to different subjects, or where the information contained in the attribute instances belongs to different entities. Row 502 further indicates that this attribute instance is to be logged every ten minutes, and that a cached value for this attribute instance may be logged, rather than obtaining a fresh value from the context server, if the effective time of the cached value is less than 30 seconds before the logging time. Row 502 further indicates that the importance level of maintaining log data for this attribute instance is six, an intermediate importance level. A high importance level may preferably be attributed to a particular attribute based on such factors as whether logging the attribute is legally required or required by a business policy, or whether the attribute has implications for the user's safety. Row 502 further indicates that logging of this attribute instance should be omitted where the current value of the attribute instance matches the last-logged value of the attribute instance. In contrast, because row 501 specifies a maximum attribute value age of 00:00:00.000, the facility must request a fresh value for the instance of the user.location attribute obtained from the GPS context server each time the attribute instance is to be logged. Row 502 indicates that the logged information for this attribute instance is to be summarized weekly, for example, by removing the rows from the log that indicate that the value of this attribute instance has changed during the previous week, and replacing them with a single row indicating the total number of times that the value of this attribute instance changed during the week. Other summarization techniques may summarize the number of times that the user was at each of a number of common locations, such as work, home, and health club. Row 502 further indicates that this attribute instance was last logged at 14:04:36.121 on 2/22/2000. Based on the contents of fields 514 and 519, this attribute instance is scheduled to be next logged at 14:14:36.121 on 2/22/2000.

Figure 6 is a flow diagram showing the steps preferably performed by the facility in order to log attribute instances. In step 601, the facility identifies the logging configuration record specifying the next logging time. By examining Figure 5, it can be determined that the logging configuration record shown in row 501 has the next logging time, which is 14:07:10.011 on 2/22/2000. In step 601, the facility further waits until this next logging time arrives. In step 602, if the value cached in the characterization module for the attribute instance specified by the logging configuration record identified in step 601 satisfies the maximum attribute value age limit specified by this logging configuration record, then the facility continues in step 604, else the facility continues in step 603. In step 603, the facility uses the characterization module to obtain a fresh value of the specified attribute instance from the specified context server. For example, the facility uses the characterization module to obtain a fresh value of the user.location attribute from the GPS context server. In a further preferred embodiment, the facility in stepa 602-603 may use a characterization module operating in a different computer system to obtain a value for the logged attribute instance. That is, the facility may log attributes obtained in computer systems other than the one in which the facility is executing.

After step 603, the facility continues in step 604. In step 604, the facility performs any pre-logging processing specified by the identified logging configuration record. Examples of pre-logging processing techniques preferably supported by the facility include summarization, in which several values of one attribute instance, or values of a number of different attribute instances are summarized in a single log record; abstraction, in which the value of one or more attribute instances, are abstracted into a higher-level conceptualization; and compression, in which information to be stored in the log record is modified to fit in a smaller space.

In step 605, if the pre-logging processing performed in step 604 obviates the current logging iteration, then the facility continues in step 607 to abort the current logging iteration, else the facility continues in step 606. In step 606, the facility stores a record for the specified attribute instance in the specified log. Those skilled in the art will appreciate that the facility may be straightforwardly adapted to support additional techniques of a similar nature. To maximize the accessibility of the data stored in the log, the log record is preferably stored as tab-delimited ASCII text or another widely-supported format. The contents of a stored log record are discussed further in conjunction with Figure 7 below. After step 606, the facility continues in step 607. In step 607, the facility updates the last logged field of the logging configuration record with the current time. After step 607, the facility continues in step 601 to identify the logging configuration record specifying the next logging time.

Figure 7 is a data structure diagram showing an attribute log table produced by the facility. The attribute log table 700 contains rows 701-704, each corresponding to a single logged value of an attribute instance. Each row contains an attribute name field 711 indicating the attribute name of the logged attribute instance; a context field server field 712 identifying the context server from which the attribute instance is obtained; a value field 713 containing the logged value for the attribute instance; an uncertainty field 714 indicating the level of uncertainty of the logged value; a time-stamp field 715 indicating the effective time at which the logged value was most accurate; a units field 716 indicating the units in which the value and uncertainty are expressed; a format version field 717 indicating the version of the format in which the value and uncertainty are expressed; a flags field 718 containing additional information about the logged attribute instance; and a logged time field 719 indicating the time at which the value of the attribute instance was logged.

For example, it can be seen from row 704 that the most recently-logged attribute instance is the instance of the user.location attribute provided by the GPS context server. The logged value of this attribute instance is 47° 38.72' N, 122° 18.22' W, with an uncertainty of 0° .10'. This attribute instance value has an effective time of 14:05:10.008 on 2/22/2000. The value and uncertainty are expressed in the units of degrees and minutes, the format version is 1.01, and there are no flags stored for attribute instance value. This attribute log record was generated at 14:05:10.011 on 2/22/2000.

Figure 8 is a flow diagram showing the steps preferably performed by the facility in order to maintain an attribute log. In steps 801-805, the facility loops through each attribute log maintenance cycle. Those skilled in the art will appreciate that the facility could be configured to perform attribute log maintenance cycles at a variety of frequencies, such as every hour, every day, or every month. In steps 802-804, the facility loops through each log configuration record that specifies a maintenance technique in the maintenance processing field 518. In step 803, the facility applies the specified maintenance technique to the log records corresponding to the logging configuration record, which may include log records stored in multiple logs and/or in multiple locations. Examples of maintenance techniques preferably supported by the facility include compression, summarization and abstraction, discussed above; thinning, in which some log records for an attribute instance are deleted from the log, while others are retained in the log; discarding, in which all values of an attribute instance during a particular time period are deleted from the log; and archiving, in which log records for a particular attribute instance are removed from the log and stored in another location, such as on a back-up storage device. Those skilled in the art will appreciate that the facility may be straightforwardly adapted to support additional techniques of a similar nature.

The facility preferably makes decisions regarding when and to which attribute instances to apply maintenance techniques based on a variety of factors, including the relative importance level specified for various attribute instances, as well as other indications of the need to retain the logged information, such as an indication that logging information for a particular attribute instance has become available from another source, such as a database accessible via the Internet.

In step 804, if additional logging configuration records remain, then the facility continues in step 802 to process the next logging configuration record. In step 805, the facility continues in step 801 to process the next context log maintenance cycle.

Figure 9 is a flow diagram showing the steps preferably performed by the facility to analyze log records for attribute instances. In step 901, the facility receives an analysis specification specifying an attribute instance, including the attribute name of the attribute instance and the context server from which the attribute instance is obtained, the time period in which to analyze the logged attribute instance, and an analysis technique. Such a specification may be received from the user via the user interface, or from an application or other program whose functionality includes logged attribute analysis.

Examples of analysis techniques preferably supported by the facility include the following: summarization, discussed above; combination, in which values of several different attribute instances from the same time period are combined; generating a new value of the context attribute that is made available via the characterization module; retrospective application of rules, in which a rule designed to be applied to real-time values of an attribute is applied to logged values of that attribute; and pattern analysis, where recurring patterns are identified in the logged attribute values, or where a pattern is identified in the logged attribute data that matches a pattern specified outside the logged attribute data -- for example, matching a series of logged electrocardiogram attribute values to an external electrocardiogram model indicating a high likelihood of heart disease; and ongoing analysis, wherein interim results produced during earlier analysis are retrieved, employed in current processing, augmented, and stored, in some cases separately from the log. Those skilled in the art will appreciate that the facility may be straightforwardly adapted to support additional techniques of a similar nature.

In step 902, the facility applies the specified analysis technique to logged values of the specified attribute instance having effective times during the specified period. After step 902, the facility continues in 901 to receive the next analysis specification.

Figure 10 is a flow diagram showing the steps preferably performed by the facility in order to conduct a context simulation. In step 1001, the facility receives a simulation specification specifying one or more attribute instances, a time period, and a speed for a new simulation. Such a specification may be received from the user via the user interface, or from an application or other program whose functionality includes logged attribute analysis. In simulation step 1002, the facility performs the simulation by causing the characterization module to, when it receives a request for one of the specified attribute instances, return a logged value for the attribute instance rather than returning a value for the attribute instance generated in real-time by the corresponding context server. (The characterization module preferably continues to respond to attribute instance requests for other attribute instances with a value for the attribute instance generated in real-time by the corresponding context server. The facility is preferably configurable, however, to disable provision of real-time values for some or all of the other attribute instances.) As the simulation advances in time, the logged times of the logged values that are returned are also advanced in accordance with the specified simulation speed. Simulations may preferably be used for compliance testing or other testing, to repeat an experience, or for other purposes. In one preferred embodiment, scheduled logging is suspended during performance of the simulation in step 1002.. After step 1002, the facility continues in step 1001 to receive the next simulation specification.

Figure 11 is a data flow diagram showing the operation of the facility without a characterization module. It can be seen in Figure 11 that context servers 1110, 1120, and 1130 provide attributes directly to context clients 1130, 1140, and 1150. For example, it can be seen that context server 1120 provides a user.elevation attribute 1122 directly to context client 1150. In this embodiment, the context client may itself cache attribute values recently obtained from a context server. Further, in this embodiment, context clients may themselves interrogate context servers for an enumeration of their attributes, and mediate between attribute instances provided by different context servers. For example, context client 1140 may mediate between the instance 1111 of the user.location attribute provided by context server 1110 and the instance 1121 of the user.location attribute provided by context server 1120.

In this embodiment, the facility preferably uses a logging context client 1160 to obtain attribute instance values from a context server and store them in a logging database 1161. For example, the diagram shows that the logging context client obtains values of the instance of the user.location attribute supplied by the gps context server and the instance of the in_region attribute supplied by the region_analysis context server and stores them in the logging database.

It will be understood by those skilled in the art that the above-described facility could be adapted or extended in various ways. For example, the facility may be implemented in computing devices other than wearable computers, including other types of mobile computing devices, such as personal digital assistants. The facility may further be implemented on various types of stationary and/or distributed computing devices, as well as non-traditional computing devices, such as smart appliances. Further, the facility may use attribute instance tables, logging configuration tables, and attribute log tables of various forms other than those shown above or may maintain its state in ways that do not involve the use of such tables. The facility may also utilize various additional pre-logging processing, log maintenance, and log analysis techniques beyond those described above. In addition to the information discussed above, additional information may be included in a log record, including the values of other attributes. Log records may preferably be ordered and/or indexed based upon any, all, or of the information contained in them.

While the foregoing description makes reference to preferred embodiments, the scope of the invention is defined solely by the claims that follow and the elements recited therein.